R E P O R T R E S U M E S

ED 012 012 AL 000 050

SOME COMIT SUBROUTINES FOR TESTING GENERATIVE GRAMMARS.

BY- YANG, JAMES
OHIO STATE UNIV., COLUMBUS, RESEARCH FOUNDATION

REPORT NUMBER POLA-13-3 PUB DATE AUG 66
EDRS PRICE MF-$0.09 HC-$0.92 23P.

DESCRIPTORS- *COMPUTATIONAL LINGUISTICS, *GRAMMAR, *COMPUTER ORIENTED PROGRAMS, *LINGUISTICS, PROJECT ON LINGUISTIC ANALYSIS, COMIT, GENERATIVE GRAMMARS, SUBROUTINES, COLUMBUS

A SERIES OF SUBROUTINES (READRULE, ANNOTATE, CHOOSE, STORERULE, ADD-ONE AND SUB-ONE, DOMINATE, SUBTREE, TERMINAL, AND EQUAL) WRITTEN UP IN THE "COMIT" PROGRAMING LANGUAGE TO PROVIDE A CONVENIENT WAY FOR LINGUISTS TO TEST ANY SET OF GRAMMAR RULES AND TO GENERATE SENTENCES, ARE GIVEN WITH EXPLANATIONS FOR EACH. THIS IS FOLLOWED BY COMPLETE LISTINGS OF THE SUBROUTINES, TOGETHER WITH A MORE TECHNICAL DESCRIPTION OF WHAT THEY DO. THE USER OF THESE ROUTINES IS NOT REQUIRED TO HAVE ANY BACKGROUND OR EXPERIENCE IN DIGITAL COMPUTER PROGRAMING. (IT)

RF Project 1685 - 6

THE OHIO STATE UNIVERSITY

RESEARCH FOUNDATION

1314 KINNEAR ROAD COLUMBUS, OHIO 43212

Project on Linguistic Analysis
Report No. 13

TOWARD A MODERN THEORY OF CASE
Charles J. Fillmore
Division of Linguistics

SUBJECT AND OBJECT IN MANDARIN
Shuan Fan Huang
Division of Linguistics

SOME PROBLEMS CONCERNING THE ENGLISH EXPLETIVE 'IT'
D. Terence Langendoen
Division of Linguistics

CORRECTIONS AND ADDITIONS TO "TREE REPRESENTATIONS IN LINGUISTICS"
Leroy F. Meyers
Department of Mathematics

SOME COMIT SUBROUTINES FOR TESTING GENERATIVE GRAMMARS
James Yang

August 1966
NATIONAL SCIENCE FOUNDATION
Grant No. GN-174

SOME COMIT SUBROUTINES FOR TESTING GENERATIVE GRAMMARS

James Yang

A series of subroutines (READRULE, ANNOTATE, CHOOSE, STORERULE, ADD-ONE and SUB-ONE, DOMINATE, SUBTREE, TERMINAL, and EQUAL) has been written up in the COMIT programming language to provide a convenient way for linguists to test any set of grammar rules and to generate sentences. The user of these routines is not required to have any background or experience in digital computer programming. When one tests the linguistic rules, all he needs to do is merely punch the rules on data cards and submit the data cards together with the relevant subroutines and a skeleton program to the computing center. Since not all subroutines necessary for this procedure have been written up, only certain types of testing can be done at present.

In general, we punch a linguistic rule in the form

LEFT-CONTEXT/LEFT-SIDE/RIGHT-CONTEXT=RIGHT-SIDE/CONDITIONS$

where the dollar sign is used to indicate to the computer the end of the rule. A rule may extend over several punched cards. If there is no context, i.e. if the rule is context-free, then both slants to the left of the equals sign are omitted. Similarly, if there are no conditions, then the slant to the right of the equals sign is omitted.

For instance, a linguist may write a context-free rule without conditions as follows:

$$S \rightarrow A \left\{ \begin{array}{l} B(C)\ (D)E^{*} \left\{ \begin{array}{l} F \\ G \end{array} \right\} \\ (H\ I\ J) \left\{ \begin{array}{l} K \\ L \\ M(N) \end{array} \right\} \end{array} \right\} \qquad (1)$$

Each letter stands for a constituent, which may actually consist of more than one written character. The notation has the following significance.

(C) means: the constituent C is optional.

E* means: the constituent E is repeatable (i.e., E, E E, E E E, etc., are alternatives).

$\begin{Bmatrix} F \\ G \end{Bmatrix}$ means: Either F or G may be chosen, but not both of them.

(H‡J) means: at least one of H, J must be used (i.e., H, J, or H J), in the order given; similarly

(H I‡J) is equivalent to $\begin{Bmatrix} H\ I \\ J \\ H\ I\ J \end{Bmatrix}$.

The linguist's two-dimensional display of the rules is compressed to a linear expression

S = A(B(C,)(D,)E*(F,G),(*H I,J)(K,L,M(N,)))$ (2)

which is punched on a card. This compression is obtained in the following way.

A comma means "or".

A star not immediately after a left parenthesis means: repeat the preceding element as often as desired.

A star immediately after a left parenthesis means: choose at least one element from what is included in the parentheses; if several elements are chosen, they must be in the order given. An element may consist of one or more constituents, and is bounded by commas or parentheses.

A dollar sign means: the end of the rule.

Thus, (C) in the linguist's display (1) is rewritten as (C,) in (2), since the alternatives are C and the empty string. Also,

E* in (1) corresponds to E* in (2), $\begin{Bmatrix} F \\ G \end{Bmatrix}$ is rewritten as (F,G) , and (H I‖J) is rewritten as (*H I,J) .

A rule in the form (2) can be easily punched on one or more data cards. Constituents in a rule are separated by spaces or punctuation marks, and optional spaces may be used (except within single constituents.

The first subroutine (READRULE) reads in data cards until a dollar sign has been found, and makes the first tests for well-formedness. This subroutine has been completed only for context-free rules without conditions. The left side is separated from the right side.

The next subroutine (ANNOTATE) in effect puts subscripts on the commas and parentheses on the right side of a linguistic rule, for easy reference when the rule is later applied. In the actual operation of the subroutine, however, the parentheses are replaced by commas, and an indication of depth of parenthesis nesting is placed as a separate constituent before each comma. An additional pair of parentheses is placed about the whole right side. Thus each comma or parenthesis is replaced by a pair of constituents (called a "marker") $i_{,j,q}$ where

i is the depth of parenthesis nesting,

j is the number of alternatives preceding this marker at this level (and so is 0 if the marker replaces a left parenthesis),

and q is a special subscript, which is END if this is the last marker of this component at this level (i.e. if the marker corresponds to a right parenthesis); the subscript includes USE if at least one element of this component must be used (i.e. if the left parenthesis for this component is immediately followed by a star).

A star not after a left parenthesis becomes a special constituent *REPEAT.

Thus the right side of (1) or (2) is annotated as

$$1\ ,_0\ A\ 2\ ,_0\ B\ 3\ ,_0\ C\ 3\ ,_1\ 3\ ,_{2,END}\ 3\ ,_0\ D\ 3\ ,_1\ 3\ ,_{2,END}\ E\ *_{REPEAT}$$
$$3\ ,_0\ F\ 3\ ,_1\ G\ 3\ ,_{2,END}\ 2\ ,_1\ 3\ ,_{0,USE}\ H\ I\ 3\ ,_{1,USE}\ J\ 3\ ,_{2,USE,END} \qquad (3)$$
$$3\ ,_0\ K\ 3\ ,_1\ L\ 3\ ,_2\ M\ 4\ ,_0\ N\ 4\ ,_1\ 4\ ,_{2,END}\ 3\ ,_{3,END}\ 2\ ,_{2,END}\ 1\ ,_{1,END}$$

The subroutines ADD-ONE and SUB-ONE are needed to perform the arithmetic in constructing the markers, since the COMIT programming language provides arithmetic only for subscripts.

An annotated rule is then stored away by the subroutine STORERULE. Control is then returned to READRULE, and another grammar rule is read in. After all grammar rules have been stored, we may apply them to generate sentences. Of course, we should specify in advance the number of sentences we want and the proper initial symbol (usualy S), which is taken as the name of the root node of the phrase marker.

The subroutine CHOOSE will choose an alternative from the annotated right side of a linguistic rule. At present, the choice is made according to the COMIT random subrule chooser. It is assumed that no choice involves more than 36 alternatives.

The subroutine CHOOSE is used to pick a number from 1 to n, where n is a positive integer less than 37, by using the random feature of COMIT. An unwritten subroutine, SUBRULE, will use this choice to pick one of the alternatives in a grammatical linguistic rule. For example, in the rule given in (3), a choice is made from the one component at level 1, yielding the whole rule. Then the constituent A is chosen automatically, and a choice is made among the two components at the second level. Suppose that the second component is chosen. We now have chosen a simpler rule

A(*H I,J)(K,L,M(N,))

in the notation of (2). Then a third-level choice is made. Suppose that we choose both H I and J. Thus we have chosen

A H I J(K,L,M(N,))

Again a third-level choice is made. Suppose that the third element M(N,) of this component is chosen. Thus we have chosen

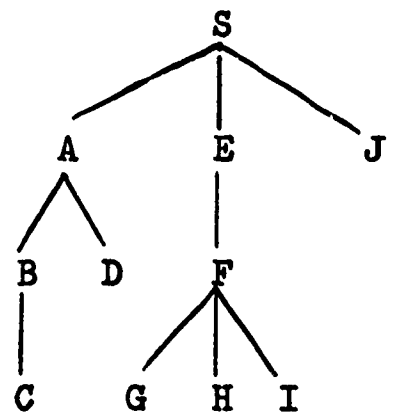
A H I J M(N,)

The constituent M is now required, and a fourth-level choice is made between N and the empty string. Suppose that the empty string is chosen. Thus we have chosen

A H I J M

which is our final rule since no more choices can be made.

A sequence of linguistic rule applications leads to a phrase-marker tree. We use modified Polish notation to indicate such a tree. Simply stated, the numerical subscript on a constituent (i.e. a node of the tree) gives the number of nodes which are dominated directly by this constituent. For example, the tree



(4)

is specified as

S_3 A_2 B_1 C D E_1 F_3 G H I J (5)

since the node S dominates the three nodes A , E , and J ; the node A dominates the two nodes B and D ; the node B dominates the one node C ; the nodes C , D , G , H , I , and J are terminal nodes (i.e. they dominate no nodes); a terminal node is considered to have a zero numerical subscript. or no numerical subscript.

Several of the subroutines are concerned with manipulation of trees. One of them, TERMINAL, constructs the string of terminal symbols from a given tree. Thus the result of applying TERMINAL to the tree (4) or (5) is

C D G H I J

The subroutine SUBTREE divides a tree into three parts according to a specified node occurrence. The middle part consists of this specified node, together with everything dependent on it. The left part and the right part are the parts of the original tree which are to the left and right of the middle part. Thus, if the subtree dominated by the node F in (4) and (5) is wanted, the result is

left: S_3 A_2 B_1 C D E_1

middle: F_3 G H I

right: J

The subroutine DOMINATE determines the node which immediately dominates a given node in a tree. A special marker $*X_1$ is placed before the given node. Thus the result of applying DOMINATE to the node D in (4) and (5) is

left: S_3

middle and right: A_2 B_1 C $*X_1$ D E_1 F_3 G H I J

The subroutine EQUAL tests whether or not two strings (such as trees) are equal.

The complete listings of the subroutines, together with a more technical description of what they do, follow.

```
(SUBROUTINE READRULE. FIRST VERSION.)
(ENTRY POINT. READRULE)
(FUNCTION.  A LINGUISTIC RULE IS GIVEN ON CARDS IN THE FORM)
(LEFT-CONTEXT / LEFT-SIDE / RIGHT-CONTEXT = RIGHT-SIDE / CONDITIONS $)
(IF NO CONTEXT, BOTH SLANTS ON LEFT ARE OMITTED.)
(IF NO CONDITIONS, THE LAST SLANT IS OMITTED.)
(A RULE IS TERMINATED BY $. ALL SYMBOLS AFTER $ ON A CARD ARE IGNORED.)
(READRULE READS IN THESE CARDS IN FORMAT C FROM THE INPUT TAPE AND)
(STORES THE LEFT SIDE AND THE RIGHT SIDE OF THE RULE ON SHELVES 111)
(AND 112, RESPECTIVELY.)
(IN THE LEFT-SIDE AND THE RIGHT-SIDE, SPACES OR PLUS SIGNS SEPARATE)
(LINGUISTIC CONSTITUENTS, COMMAS SEPARATE ALTERNATIVES, AND PARENTHESES)
(ENCLOSE SETS OF ALTERNATIVES.  A STAR AFTER A LEFT PARENTHESIS)
(INDICATES THAT AT LEAST ONE ALTERNATIVE--POSSIBLY SEVERAL ALTERNATIVES)
(IN A ROW--MUST BE CHOSEN.  A STAR OTHERWISE INDICATES THAT THE ITEM)
(BEFORE IT  MAY BE REPEATED.  AN INITIAL AND FINAL SET OF PARENTHESES)
(MAY BE OMITTED.)
(EACH LINGUISTIC CONSTITUENT IS COMPRESSED INTO A SINGLE COMIT)
(CONSTITUENT.  EACH PARENTHESIS, COMMA, AND STAR IS COPIED AS A SINGLE)
(COMIT CONSTITUENT.  THE SPACE AND PLUS SIGN ARE REMOVED.  SOME REMOVAL)
(OF EXTRA PARENTHESES, SPACES, AND COMMAS IS DONE IN THE LEFT-SIDE.)
(THE RESULT OF THIS TRIMMING IS LEFT IN SHELF 111.  IF THE LEFT-SIDE)
(HAS ONLY ONE LINGUISTIC CONSTITUENT, THIS IS COMPRESSED INTO A SINGLE)
(COMIT CONTITUENT.)
(AT PRESENT,THIS SUBROUTINE CAN DEAL ONLY WITH RULES HAVING NO)
(CONTEXT AND NO CONDITIONS,WITH ONLY ONE LEFT-SIDE CONSTITUENT.)
(RESTRICTION.  THE SYMBOLS  + , $ = / *  PARENTHESES AND SPACE MAY NOT)
(BE USED WITHIN LINGUISTIC CONSTITUENTS.)
(WORKSPACE. INPUT,DESTROYED. OUTPUT,RETURN)
(SHELVES USED. 111,OUTPUT,LEFT HALF)
(              112,OUTPUT,RIGHT HALF)
(              127,PUSHDOWN STORE FOR SUBROUTINES)
(EXIT. 1. SUCCESSFULLY READ IN ONE GRAMMAR RULE.)
(      2. FAILURE. THE RULE IS CONTEXT-SENSITIVE.)
(       PRINT. IN SUBROUTINE READRULE, THE RULE +WHOLERULE+ IS CONTEXT-)
(              SENSITIVE.)
(      3. FAILURE. THE RULE IS LEFT-COMPOUND.)
(      PRINT. IN SUBROUTINE READRULE, THE LEFT SIDE OF THE RULE)
(              +WHOLE RULE+ IS COMPOUND.)
(      4. FAILURE. THE RULE HAS CONDITIONS.)
(         PRINT. IN SUBROUTINE READRULE, THE RULE+WHOLE RULE+HAS )
(              CONDITIONS.)
(      5. FAILURE. THE RULE HAS NO = SIGN.)
(         PRINT. IN SUBROUTINE READRULE, THE RULE+WHOLE RULE+HAS NO)
(              EQUALS SIGN.)
(      6. FAILURE. THE RULE HAS NO END SYMBOL $.)
(         PRINT. IN SUBROUTINE READRULE, THE END OF RULE $ IS NEVER)
(              FOUND IN+WHOLEDATA+.)
(IN EXITS 5 AND 6, STORE ALL IN SHELF 111.  OTHERWISE, STORE LEFT HALF)
(AND RIGHT HALF SEPARATELY.  IN EXITS 1, 3, AND 4, THE LEFT SIDE IS)
(TRIMMED, AND IN EXITS 1 AND 4, IT IS COMPRESSED.)
(SUBROUTINE READRULE HAS 36 COMIT RULES.)
READRULE  $ = //*A111 1,*A112 1,*A112 1,*RCK1  *
READRULE.0  $+*$+$+*. = A+1 //*A112 1  READRULE.1
*  $+*. = //*Q112 1,*RCK2  READRULE.0
*  $ = //*A112 1  *
*  $ = 1+-IN-SUBROUTINE-READRULE,-THE-END-OF-RULE-*$-IS-NEVER-FOUND-
-IN*.-+1+*.*0  //*Q111 1,*WAM2 3 4,*N127 1  *
*  $1 = 1/.6  $ (NO SYMBOL $,END OF FILE)
READRULE.1  $+*=+$ = 1+3//*Q112 2  READRULE.2
*  $ = -IN-SUBROUTINE-READRULE,-THE-RULE*.-+1+*$*.-HAS-NO-EQUALS-SIGN.-
*.*0+1 //*WAM1 2 3,*N127 1,*Q111 4  *
*  $1 = 1/.5  $ (NO EQUALS SIGN)
READRULE.2  $+*/+$ = A+1+2+3+* +A //*N127 1,*A112 6  READRULE.3
```

```
*  $ = *A+1+*B  READRULE.4
READRULE.3  $1+$+*=+$ = 1/.2+2+4+-IN-SUBROUTINE-READRULE,-THE-RULE*.-+2-
+3+4+*$*.-IS-CONTEXT*-SENSITIVE.*.*0 //*Q111 2,*Q112 3,*WAM4 5 6 7 8  $
     (CONTEXT-SENSITIVE)
READRULE.4  *A+- = 1  READRULE.4
READRULE.4A  *A+*+ =1  READRULE.4
READRULE.5  -+*B = 2  READRULE.5
READRULE.5A  *+ +*B=2  READRULE.5
READRULE.5B  *A+*(+$+*)+*B = 1+3+5  READRULE.4
*  -  =  READRULE.6
*  ,  =  READRULE.6
*  *(  =  READRULE.6
*  *)  =  READRULE.6
*  *+  =  READRULE.6
*  **  =  READRULE.6
*  *A+$+*B = 2//*K1,*Q111 1,*A112 1  READRULE.7
READRULE.6  *A+$+*B = A+2+*=+B //*N127 1,*A112 4  *
*  $1+$+*=+$ = 1/.3+2+4+-IN-SUBROUTINE-READRULE,-THE-LEFT-SIDE-OF-THE--
RULE*.-+2+3+4+*$*.-IS-COMPOUND.*.*0 //*Q111 2,*Q112 3,*WAM4 5 6 7 8  $
     (LEFT-COMPOUND)
READRULE.7  $+*/+$ = A+A+*=+1+2+3  //*N127 1,*A111 2  READRULE.11
*  $ = *A+1  *
READRULE.8  *A+$1 = //*L2  READRULE.LIS
*  $+*A = //*Q112 1,*N127 2  *
*  $1 = 1/.1  $ (SUCCESS)
-READRULE.LIS  -  = 0  READRULE.9
               *+ = 0  READRULE.9
               *(  =  READRULE.10
               *)  =  READRULE.10
               ,  =  READRULE.10
               **  =  READRULE.10
*  $+*A+$1 = 1+3+2  //*K1 2  READRULE.8
READRULE.9  $+*A = //*Q112 1  READRULE.8
READRULE.10  $+*A+$1 = //*Q112 1 3  READRULE.8
READRULE.11  $1+$+*=+$ = 1/.4+2+4+-IN-SUBROUTINE-READRULE,-THE-RULE*.-+-
2+3+4+*$*.-HAS-CONDITIONS.*.*0 //*Q111 2,*Q112 3,*WAM4 5 6 7 8 $(CONDS.)
(END OF SUBROUTINE READRULE.)
```

```
(SUBROUTINE ANNOTATE.FIRST VERSION)
(ENTRY POINT. ANNOTATE)
(FUNCTION.  FOR RIGHT SIDE, AFTER READRULE.  REPLACES ALL PARENTHESES,)
(COMMAS, AND STARS BY COMMA WITH SUBSCRIPTS, TOGETHER WITH AN)
(INDICATION OF DEPTH OF PARENTHESIS NESTING. OTHER SYMBOLS--THE)
(NODE NAMES--ARE LEFT UNCHANGED. THE COMMA WITH SUBSCRIPTS)
(WILL BE USED LATER FOR RANDOM CHOICE OF ALTERNATIVE SUBRULES.)
(EACH LEFT PARENTHESIS, RIGHT PARENTHESIS, OR COMMA IS REPLACED BY)
(A MARKER *M+,/.N,Q WITH TWO CONSTITUENTS, WHERE M IS THE )
(DEPTH OF PARENTHESIS NESTING, N IS THE NUMBER OF ALTERNATIVE)
(CONSTITUENT STRINGS PRECEDING THIS MARKER IN THIS COMPONENT)
(AT THIS LEVEL, AND Q IS A LOGICAL SUBSCRIPT, WHICH DOES NOT)
(ALWAYS APPEAR.)
(N IS 0 FOR A LEFT PARENTHESIS.)
(Q INCLUDES END FOR RIGHT PARENTHESIS.)
(EACH STAR IS REPLACED BY **/REPEAT  IF IT IS NOT IMMEDIATELY AFTER)
(A LEFT PARENTHESIS. OTHERWISE,IT MEANS CHOOSE AT LEAST ONE OF)
(THE CONSTITUENT STRINGS IN THIS COMPONENT,AND IS DELETED,)
(BUT  USE  IS PUT AS SUBSCRIPT ON ALL MARKERS *M+,/.N,Q IN THIS)
(COMPONENT AT THIS LEVEL.)
(THE ORIGINAL STRING IS ENCLOSED IN PARENTHESES INITIALLY.)
(ALL OTHER INPUT SYMBOLS REMAIN UNCHANGED.)
(RESTRICTIONS.  NONE.)
(SUBROUTINE USED.  ADD-ONE.)
(WORKSPACE.  INPUT.  ORIGINAL STRING.)
(            OUTPUT.  RETURN FOLLOWED BY ANNOTATED STRING.)
(SHELVES USED. 123,TEMPORARY STORAGE FOR ORIGINAL UNANNOTATED STRING)
(              124,TEMPORARY STORAGE FOR UNFINISHED PART)
(              125,TEMPORARY PUSHDOWN STORAGE FOR SUBSCRIPTING)
(              126,TEMPORARY STORAGE FOR FINISHED PART)
(              127,PUSHDOWN STORE FOR SUBROUTINES)
(EXIT. 1.  SUCCESS. THE WORKSPACE CONTAINS THE ANNOTATED STRING)
(                  PRECEDED BY THE SUBROUTINE RETURN,)
(          AND SHELF 123 CONTAINS THE ORIGINAL STRING.)
(      2.  FAILURE. THE INPUT STRING IS NOT WELL FORMED.)
(          SHELF 126 CONTAINS THE PORTION ALREADY ANNOTATED.)
(          PRINT. IN SUBROUTINE ANNOTATE, THE STRING +WHOLE)
(                 STRING+ IS NOT WELL FORMED.)
(      3.  FAILURE. THE INPUT STRING IS EMPTY.)
(          PRINT. IN SUBROUTINE ANNOTATE, THE INPUT STRING IS EMPTY.)
(IN CASE OF FAILURE,THE ORIGINAL INPUT IS RESTORED TO THE WORK-)
(SPACE,PRECEDED BY THE SUBROUTINE RETURN.)
(SUBROUTINE ANNOTATE CONTAINS 22 RULES.)
ANNOTATE  $1+$ = A+*0+*(+1+2+*)+1+2 //*A123 1,*A124 1,*A125 1,*A126 1,-
  *A126 1,*S125 2,*Q124 4 5 6,*Q123 7 8,*L3  ANNOTATE.L
*  $=-IN-SUBROUTINE-ANNOTATE,-THE-INPUT-STRING-IS-EMPTY.*.*0-
      //*WAM1,*N127 1  *
*  $1 = 1/.3  $ (EMPTY WORKSPACE)
-ANNOTATE.L  *(= //*N124 1  ANNOTATE.1
             , = A+A//*N125 1,*N125 2  ANNOTATE.5
             *)=  A+A//*N125 1,*N125 2  ANNOTATE.7
             ** = **/REPEAT //*Q126 1  ANNOTATE.0
*  $1= //*Q126 1 *
ANNOTATE.0  $=//*N124 1  *
*  $1= //*L1  ANNOTATE.L
*  $=  //*A125 1  *
*  $2 =  ANNOTATE.8
*  $=A+A//*N127 1,*A126 2  *
*  $1=1/.1  $(SUCCESS)
ANNOTATE.1  ** = ,/.0,USE  ANNOTATE.2
*  $1=1+,/.0 //*S124 1  *
ANNOTATE.2  $=ADD-ONE/ANNOTATE.3+1//*S127 1,*S126 2,*N125 1  *
*  $1=1+1//*S125 1,*E2  ADD-ONE
ANNOTATE.3  $1+$=2+A  //*N126 2,*K1  *
```

```
ANNOTATE.5  $1+$1=1+2+1+2/.11//*Q126 1 2,*S125 4 3  ANNOTATE.0
*  ANNOTATE.8
ANNOTATE.7  $1+$1=1+2/END //*Q126 1 2  ANNOTATE.0
ANNOTATE.8  $ = //*A123 1  *
*  $ = -IN-SUBROUTINE-ANNOTATE,-THE-STRING*.+1+-IS-NOT-WELL*-FORMED.*,-
*0+1 //*WAM1,*WSM2,*WAM3,*N127 1  *
*  $1=1/.2  $ (NOT WELL-FORMED)
(END OF SUBROUTINE ANNOTATE)
```

```
(SUBROUTINES  ADD-ONE  AND  SUB-ONE .)
(ENTRY POINTS.  ADD-ONE  AND  SUB-ONE.)
(IF THE WORKSPACE CONSISTS OF A STRING OF SINGLE-DIGIT CONSTITUENTS)
(ONLY, THEN 1 IS ADDED TO THE DECIMAL NUMBER IT REPRESENTS, FOR)
(ADD-ONE, OR 1 IS SUBTRACTED FROM THE DECIMAL NUMBER IT REPRESENTS,)
(FOR SUB-ONE.)
(SUPERFLUOUS INITIAL ZEROS ARE ELIMINATED.)
(CONSTITUENTS RETAIN THE SUBSCRIPTS OF THE DIGITS THEY REPLACE.)
(THUS, ADD-ONE APPLIED TO   *0/A + *0/B +*9/C + *9/.5  )
(                  YIELDS           *1/B +*0/C + *0/.5 .)
(RESTRICTIONS.  NONE.)
(WORKSPACE.  INPUT.  ORIGINAL STRING.)
(            OUTPUT.  RETURN FOLLOWED BY THE RESULTING STRING OF)
(                     ONE-DIGIT CONSTITUENTS.)
(SHELVES USED.  127.  PUSHDOWN STORE FOR SUBROUTINES.)
(EXITS 1. SUCCESSFUL ADDITION OR SUBTRACTION.)
(      2. FAILURE.  THE WORKSPACE IS INITIALLY EMPTY.)
(           PRINT.  IN SUBROUTINE ADD-ONE  OR  SUB-ONE ,)
(           THE WORKSPACE IS EMPTY.)
(      3. FAILURE.  THE WORKSPACE CONTAINS THINGS OTHER THAN DIGITS.)
(           PRINT.  IN SUBROUTINE ADD-ONE  OR  SUB-ONE, THE WORKSPACE,)
(           + STRING + DOES NOT CONTAIN ONLY DIGITS. )
(      4. FAILURE FOR SUB-ONE ONLY.  THE WORKSPACE CONTAINS ZERO ONLY.)
(           PRINT.  IN SUBROUTINE SUB-ONE, THE WORKSPACE CONTAINS )
(           ZERO ONLY. )
(IN ALL FAILURE EXITS, THE WORKSPACE IS RESTORED, PRECEDED BY THE RET.)
(SUBROUTINES   ADD-ONE  AND  SUB-ONE  HAVE 21 RULES ALTOGETHER.)
ADD-ONE  $ = -IN-SUBROUTINE-ADD*-ONE, +1 //*S127 1,ADD-ONE.2 ADD,-
       ADD-ONE.3 ADD ADD-ONE.0
SUB-ONE  $ = -IN-SUBROUTINE-SUB*-ONE, +1 //*S127 1,ADD-ONE.2 SUB,-
       ADD-ONE.3 SUB  *
ADD-ONE.0  $1 = *Z+1  ADD-ONE.1
*  $ = A+A //*N127 2,*N127 1  *
*  $1+$1 = 1/.2+2+-THE-WORKSPACE-IS-EMPTY.*. //          *WAM2 3 $(EMPTY)
ADD-ONE.1  *Z+$1 = 2+*Z/$*2 //*L1  ADD-ONE.LIST
ADD-ONE.2 ADD  $1+*Z =*Z/$*1+1 //*L2  ADD-ONE.MIST
          SUB        =*Z/$*1+1 //*L2  SUB-ONE.MIST
ADD-ONE.3 ADD  *Z =*Z+*1  ADD-ONE.7
          SUB     = 0  *
SUB-ONE.4  *9+$ = 2+*0/$*1  SUB-ONE.4
*  $ = A+A+1 //*N127 2,*N127 1  *
*  $1+$1 = 1/.4+2+-THE-WORKSPACE-CONTAINS-ZERO-ONLY.*. //*WAM2 3 $(ZERO)
ADD-ONE.5  $+*Z+$1 = *Z+1+3/$*2  *
ADD-ONE.6  *Z+*0+$1 = 1+3 ADD-ONE.6
ADD-ONE.7  $1 = //*N127 1,*N127 1  *
*  $1 = 1/.1  $ (SUCCESS)
ADD-ONE.8  *Z+$1 = *Z+2/$*1  ADD-ONE.2
-ADD-ONE.LIST    *0 =        ADD-ONE.1
                 *1 =        ADD-ONE.1
                 *2 =        ADD-ONE.1
                 *3 =        ADD-ONE.1
                 *4 =        ADD-ONE.1
                 *5 =        ADD-ONE.1
                 *6 =        ADD-ONE.1
                 *7 =        ADD-ONE.1
                 *8 =        ADD-ONE.1
                 *9 =        ADD-ONE.1
*  $+*Z   = A+A+1   //*N127 2,*N127 1  *
*  $1+$1+$ = 1/.3+2+-THE-WORKSPACE,*.+3+-DOES-NOT-CONTAIN-ONLY-
-DIGITS.*.+3 //*WAM2 3,*WSM4,*WAM5  $ (NOT ONLY DIGITS)
-ADD-ONE.MIST  *0   = *1  ADD-ONE.5
               *1   = *2  ADD-ONE.5
               *2   = *3  ADD-ONE.5
               *3   = *4  ADD-ONE.5
```

```
                  #4    = #5   ADD-ONE.5
                  #5    = #6   ADD-ONE.5
                  #6    = #7   ADD-ONE.5
                  #7    = #8   ADD-ONE.5
                  #8    = #9   ADD-ONE.5
                  #9    =     #0   ADD-ONE.8
-SUB-ONE.MIST     #1    = #0   ADD-ONE.5
                  #2    = #1   ADD-ONE.5
                  #3    = #2   ADD-ONE.5
                  #4    = #3   ADD-ONE.5
                  #5    = #4   ADD-ONE.5
                  #6    = #5   ADD-ONE.5
                  #7    = #6   ADD-ONE.5
                  #8    = #7   ADD-ONE.5
                  #9    = #8   ADD-ONE.5
                  #0    =     #9   ADD-ONE.8
(END OF SUBROUTINES ADD-ONE AND SUB-ONE.)
```

```
(SUBROUTINE STORERULE.FIRST VERSION. FOR CONTEXT FREE RULES ONLY.)
(ENTRY POINT.STORERULE.)
(FUNCTION.  THE LEFT SIDE OF AN ANNOTATED RULE IS GIVEN ON SHELF 111.)
(AND THE RIGHT SIDE ON SHELF 112.  STORERULE PUTS THE RIGHT SIDE OF THE)
(RULE ON SHELF  N, AND QUEUES THE LEFT SIDE, FOLLOWED BY A CONSTITUENT)
( *$/.N , ON SHELF 99, WHERE  N  IS THE NUMERICAL SUBSCRIPT ON THE )
(FIRST CONSTITUENT OF SHELF 99, AND IS IN THE RANGE 1 TO 98 INCLUSIVE.)
(THIS NUMERICAL SUBSCRIPT IS THEN INCREASED BY 1 AND THEN RESTORED TO)
(THE BEGINNING OF SHELF 99.)
(NO RESTRICTION ON INPUT CONSTITUENTS.)
(WORKSPACE.INPUT DESTROYED.OUTPUT IS SUBROUTINE RETURN.)
(SHELVES USED. 1 TO 98, AS NEEDED, RIGHT SIDES STORED)
(               99, STORAGE OF MARKER *$/.N, WHERE N IS THE NEXT)
(                   SHELF TO BE USED IN STORING A RULE, AND OF)
(                   LEFT SIDES OF RULES, FOLLOWED BY MARKERS *$/.M .)
(               111, INPUT. ANNOTATED LEFT SIDE)
(               112, INPUT. ANNOTATED RIGHT SIDE)
(               127,PUSH DOWN STORE FOR SUBROUTINES)
(EXITS. 1. SUCCESS.THE RULE IS STORED ON PROPER SHELF.)
(       2. FAILURE.  THE FIRST CONSTITUENT OF SHELF 99 DOES NOT HAVE A)
(             NUMERICAL SUBSCRIPT FROM 1 TO 98 INCLUSIVE.)
(         PRINT.  IN SUBROUTINE STORERULE, THE FIRST CONSTITUENT)
(         +CONSTITUENT+  OF SHELF 99 DOES NOT HAVE A NUMERICAL SUB-)
(         SCRIPT FROM 1 TO 99 INCLUSIVE.)
(       3. FAILURE.  SHELF 99 IS EMPTY.)
(         PRINT.  IN SUBROUTINE STORERULE, SHELF 99 IS EMPTY.)
(IN CASE OF FAILURE, ALL SHELVES ARE RESTORED.)
(SUBROUTINE STORERULE HAS 7 RULES.)
STORERULE  $ = //*N99 1  *
*  $1/.L99,.GO = 1/.I1+*$/.*1 //*S99 1,*A*2 1,*A112 1,*Q*2 1,*A111 1,-
  *Q99 1 2,*N127 1  STORERULE.1
*  $1 = 1+1 //*S99 1,*N127 1  STORERULE.2
*  $ = -IN-SUBROUTINE-STORERULE,-SHELF-*9*9-IS-EMPTY.*.//*WAM1,*N127 1 *
*  $1 = 1/.3  $ (SHELF 99 IS EMPTY)
STORERULE.1  $1 = 1/.1  $ (SUCCESS)
STORERULE.2  $1+$ = 1/.2+-IN-SUBROUTINE-STORERULE,-THE-FIRST-CONSTITU-
ENT,*.+2+-OF-SHELF-*9*9-DOES-NOT-HAVE-A-NUMERICAL-SUBSCRIPT-FROM-*1-TO--
*9*8-INCLUSIVE.*. //*WAM2,*WSM3,*WAM4  $ (IMPROPER SUBSCRIPT)
(END OF SUBROUTINE STORERULE)
```

```
(SUBROUTINE CHOOSE)
(ENTRY POINT. CHOOSE)
(FUNCTION. CHOOSE AN INTEGER FROM 1 TO N, WHERE N IS THE)
(NUMERICAL SUBSCRIPT ON THE FIRST CONSTITUENT OF THE WORKSPACE.)
(AND N IS BETWEEN 1 AND 36, INCLUSIVE. THIS INTEGER N WILL APPEAR)
(AS THE NUMERICAL SUBSCRIPT ON AN EXTRA FIRST CONSTITUENT *X.)
(RESTRICTIONS. NONE.)
(WORKSPACE. INPUT, ORIGINAL STRING.)
(           OUTPUT, RETURN FOLLOWED BY ANSWER.)
(SHELVES USED. 126, TEMPORARY STORAGE.)
(              127, PUSHDOWN STORE FOR SUBROUTINES.)
(EXITS. 1. SUCCESS. A NUMERICAL SUBSCRIPT IS SUCCESSFULLY CHOSEN.)
(       2. FAILURE. THE GIVEN INTEGER IS ZERO.)
(          PRINT. IN SUBROUTINE CHOOSE, THE NUMERICAL SUBSCRIPT ON THE)
(          FIRST CONSTITUENT +CONSTITUENT+ IS ZERO.)
(       3. FAILURE. THE INTEGER IS GREATER THAN 36.)
(          PRINT. IN SUBROUTINE CHOOSE, THE NUMERICAL SUBSCRIPT ON THE)
(          FIRST CONSTITUENT +CONSTITUENT+ IS GREATER THAN 36.)
(       4. FAILURE. THE GIVEN CONSTITUENT HAS NO NUMERICAL SUBSCRIPT.)
(          PRINT. IN SUBROUTINE CHOOSE, THE FIRST CONSTITUENT)
(          +CONSTITUENT+ HAS NO NUMERICAL SUBSCRIPT.)
(       5. FAILURE. THE WORKSPACE IS EMPTY.)
(          PRINT. IN SUBROUTINE CHOOSE, THE WORKSPACE IS EMPTY.)
(IN ALL ERROR EXITS, THE WORKSPACE CONTAINS THE EXIT INFORMATION)
(FOLLOWED BY THE ORIGINAL WORKSPACE STRING.)
(SUBROUTINE CHOOSE CONTAINS 18 RULES.)
CHOOSE  $1+$ = A+*X/.*1+1+1+2 //*A126 1,*A126 1,*Q126 3 4 5  CHOOSE.1
*  $=-IN-SUBROUTINE-CHOOSE,-THE-WORKSPACE-IS-EMPTY.*.//-
     *WAM1,*N127 1  *
*  $1=1/.5  $(EMPTY)
CHOOSE.1  $1/.GO=  CHOOSE.2
*  $1/.0=  CHOOSE.5
*  $1=-IN-SUBROUTINE-CHOOSE,-THE-FIRST-CONSTITUENT*.+A+-
 -HAS-NO-NUMERICAL-SUBSCRIPT.*.-//*N126 2,*WAM1,*WSM2,*WAM3,-
     *N127 1,*A126 2  *
*  $1=1/.4  $(NO NUMERICAL SUBSCRIPT)
CHOOSE.2  $1/.G36=-IN-SUBROUTINE-CHOOSE,-THE-NUMERICAL-SUBSCRIPT-
-ON-THE-FIRST-CONSTITUENT*.+A+-IS-GREATER-THAN-*3*6.*.- //*N126 2,-
  *WAM1,*WSM2,*WAM3,*A126 2,*N127 1  CHOOSE.7
*  $1=1/CHOOSE.0+1+A/CHOOSE.4//*S126 2,CHOOSE.0  $
CHOOSE.4  $1+$1+$1=2//*D1  *
*  $1=A/CHOOSE.0+A/CHOOSE.6  $
CHOOSE.5  $=A+A//*N127 1,*A126 2  *
*  $1+$1 = 1/.2+-IN-SUBROUTINE-CHOOSE,-THE-NUMERICAL-SUBSCRIPT-ON-THE-
-FIRST-CONSTITUENT*.+2+-IS-ZERO.*.- //*WAM2,*WSM3,*WAM4  $(0 SUBSCRIPT)
CHOOSE.6  $2+$1=2+A//*A126 2  *
*  $1+$1+$1=A+2/.*1 //*N127 1  *
*  $1 = 1/.1  $ (SUCCESS)
CHOOSE.7  $1=1/.3  $ (N IS GREATER THAN 36)
CHOOSE.0  1 $1+$1=2+A/CHOOSE.0 1+A/.1  $
          2       =2+A/CHOOSE.0 1 2+A/.2  $
          3       =2+A/CHOOSE.0 1 2 3+A/.3  $
          4       =2+A/CHOOSE.0 1 2 3 4+A/.4  $
          5       =2+A/CHOOSE.0 1 2 3 4 5+A/.5  $
          6       =2+A/CHOOSE.0 1 2 3 4 5 6+A/.6  $
          7       =2+A/CHOOSE.0 1 2 3 4 5 6 7+A/.7  $
          8       =2+A/CHOOSE.0 1 2 3 4 5 6 7 8+A/.8  $
          9       =2+A/CHOOSE.0 1 2 3 4 5 6 7 8 9+A/.9  $
          10      =2+A/CHOOSE.0 1 2 3 4 5 6 7 8 9 10+A/.10  $
          11      =2+A/CHOOSE.0 1 2 3 4 5 6 7 8 9 10 11+A/.11  $
  12  =2+A/CHOOSE.0 1 2 3 4 5 6 7 8 9 10 11 12+A/.12  $
  13  =2+A/CHOOSE.0 1 2 3 4 5 6 7 8 9 10 11 12 13+A/.13  $
  14  =2+A/CHOOSE.0 1 2 3 4 5 6 7 8 9 10 11 12 13 14+A/.14  $
  15  =2+A/CHOOSE.0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15+A/.15  $
```

```
  16  =2+A/CHOOSE.0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16+A/.16  $
  17  =2+A/CHOOSE.0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17+A/.17  $
  18  =2+A/CHOOSE.0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18+-
A/.18  $
  19  =2+A/CHOOSE.0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19+-
A/.19  $
  20  =2+A/CHOOSE.0- 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 -
        21+A/.20  $
  21  =2+A/CHOOSE.0- 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 -
        +A/.21  $
  22  =2+A/CHOOSE.0- 36 35 34 33 32 31 30 29 28 27 26 25 24 23 +A/.22  $
  23  =2+A/CHOOSE.0- 36 35 34 33 32 31 30 29 28 27 26 25 24 +A/.23  $
  24  =2+A/CHOOSE.0- 36 35 34 33 32 31 30 29 28 27 26 25+A/.24  $
  25  =2+A/CHOOSE.0- 36 35 34 33 32 31 30 29 28 27 26+A/.25  $
  26  =2+A/CHOOSE.0- 36 35 34 33 32 31 30 29 28 27+A/.26  $
  27  =2+A/CHOOSE.0- 36 35 34 33 32 31 30 29 28+A/.27  $
  28  =2+A/CHOOSE.0- 36 35 34 33 32 31 30 29+A/.28  $
  29  =2+A/CHOOSE.0- 36 35 34 33 32 31 30+A/.29  $
  30  =2+A/CHOOSE.0- 36 35 34 33 32 31+A/.30  $
  31  =2+A/CHOOSE.0- 36 35 34 33 32+A/.31  $
  32  =2+A/CHOOSE.0- 36 35 34 33+A/.32  $
  33  =2+A/CHOOSE.0- 36 35 34+A/.33  $
  34  =2+A/CHOOSE.0- 36 35+A/.34  $
  35  =2+A/CHOOSE.0- 36+A/.35  $
  36  =2+A/CHOOSE.0-+A/.36  $
(END OF SUBROUTINE CHOOSE.)
```

```
(SUBROUTINE SUBTREE.)
(ENTRY POINT.SUBTREE)
(FUNCTION. IF A TREE IS GIVEN IN SHELVES 102,103,AND 104, AND IF A )
(NODE NAME IS SPECIFIED AND STORED IN SHELF 100,THEN THE SUBROUTINE)
(FINDS THE FIRST COPY OF THIS SPECIFIED NODE FROM SHELVES 103 AND 104,)
(AND DIVIDES THE TREE INTO THREE PARTS,LEFT,MIDDLE,AND RIGHT ACCORDING)
(TO THIS SPECIFIED NODE. THE MIDDLE PART CONSISTS OF THE NODE JUST)
(FOUND TOGETHER WITH ALL NODES IN THE SUBTREE DEPENDENT ON THIS)
(NODE AND IS STORED IN SHELF 103. THE LEFT AND RIGHT PARTS ARE)
(THE REMAINDER OF THE TREE AND ARE STORED IN SHELVES 102 AND 104,)
(RESPECTIVELY.  SHELF 100 IS RESTORED.  IN THE TREE,--BUT NOT ON SHELF)
(100,-- NODES WITHOUT NUMERICAL SUBSCRIPT AND THOSE WITH NUMERICAL)
(SUBSCRIPT ZERO ARE CONSIDERED EQUIVALENT.)
(RESTRICTION.NO INPUT CONSTITUENT IS       *Y OR *Z.)
(WORKSPACE.INPUT,DESTROYED. OUTPUT,SUBROUTINE RETURN.)
(SHELVES USED.100,SPECIFIED NODE STORE)
(              102,LEFT PART)
(              103,MIDDLE PART AND SUBTREE)
(              104,RIGHT PART)
(              127,PUSH DOWN STORE FOR SUBROUTINES.)
(EXIT.1.SUCCESS.SUBTREE FOUND.)
(      2.FAILURE.THE SPECIFIED NODE DOES NOT OCCUR IN THE STRING.)
(        PRINT.IN SUBROUTINE SUBTREE,+ THE SPECIFIED NODE+ DOES NOT)
(        OCCUR IN THE STRING+WHOLE STRING.)
(      3.FAILURE.SHELF 100 IS EMPTY.)
(        PRINT.  IN SUBROUTINE SUBTREE, SHELF 100 IS EMPTY.)
(      4.FAILURE.THE STRING IS NOT WELL FORMED.)
(        PRINT.IN SUBROUTINE SUBTREE,THE PORTION OF THE STRING+STRING+ )
(         BEGINNING WITH+NODE+IS NOT WELL FORMED.)
(THE PRINTOUT USES A CONSTITUENT *Y AS A SEPARATOR BETWEEN SHELVES 103)
(AND 104.)
(IN CASE OF FAILURE,SHELVES 100,102,103,AND 104 ARE RESTORED.)
(THE SUBROUTINE HAS 20 RULES.)
SUBTREE  $=//*N100 1  *
*  $1=1+*Y/.0+L+*Y/.1+L//*A103 3,*A104 5  SUBTREE.0
*  $=-IN-SUBROUTINE-SUBTREE,-SHELF-*1*0*0-IS-EMPTY.*.//*WAM1,-
     *N127 1  *
*  $1=1/.3  $ (EMPTY)
SUBTREE.0  $1/.0+*Y/.0 = 1/-.+2  SUBTREE.1
*  $1+*Y+$+1 = 1+*Z/.0,.I.*4+2+3+4 //*S100 1,*Q102 3 4,*Q103 5 SUBTREE.2
SUBTREE.0A  $1+*Y+$+*Y+$ =-IN-SUBROUTINE-SUBTREE,+1+-DOES-NOT-OCCUR-IN-
-THE-STRING-+3+*Y+5+ *.+1+3+5//*WAM1,*WSM2,*WAM3,-
      *WSM4 5 6,*WAM7,*S100 8,*Q103 9,*Q104 10,*N127 1  *
*  $1=1/.2  $ (SEARCH FAILS)
SUBTREE.1  $1+*Y+$+1 = //*Q104 3  SUBTREE.1A
*  $1+$1+$ = 1/.0+2+A+3 //*A104 3  SUBTREE.0A
SUBTREE.1A  $1+*Y/.0+$1/.GO = //*Q104 3  SUBTREE.1
*  $1+*Y+$1= 1/.0+*Z/.0       +2+A+3//*S100 1,*A104 4,*Q102 3 4,*Q103 5 *
SUBTREE.2  *Z/.GO+$1=1/.I.*2,.D1+2//*Q103 2  SUBTREE.2
*  *Z/.0  =A+*Z+B+*Z  //          *A102 1,*A103 3  SUBTREE.3
*  *Z     =A+A+A//*N100 1,*A102 2,*A103 3  *
*  $1+$+*Y+$+*Y+$=1+2+4+6+-IN-SUBROUTINE-SUBTREE,-THE-PORTION-OF-THE-
-STRING-+4+*Y+6+-BEGINNING-WITH-+1+-IS-NOT-WELL-FORMED.*.-
     //*S100 1,*Q102 2,*Q103 3,*Q104 4,*WAM5,*WSM6 7 8,*WAM9,-
     *WSM10,*WAM11,*N127 1  *
*  $1=1/.4  $ (NOT WELL FORMED)
SUBTREE.3  *Y+$+*Y = 2  *
*  $+*Z+$+*Z+$=1+3+5 //*Q102 1,*Q103 2,*Q104 3,*N127 1  *
*  $1=1/.1  $ (SUBTREE FOUND)
(END OF SUBROUTINE SUBTREE.)
```

```
(SUBROUTINE DOMINATE.)
(ENTRY POINT.DOMINATE)
(FUNCTION. SUPPOSE THAT A SENTENCE TREE IN MODIFIED POLISH NOTATION)
(IS GIVEN IN SHELVES 102,103,AND 104.THIS SUBROUTINE FINDS THE)
(NODE WHICH IMMEDIATELY DOMINATES THE FIRST CONSTITUENT OF SHELF)
(103,AND THEN BREAKS UP THE SENTENCE TREE INTO THREE PARTS SO THAT)
(THE NODE JUST FOUND, TOGETHER WITH SUCCEEDING NODES, IS REMOVED FROM)
(SHELF 102 AND PLACED AT THE BEGINNING OF SHELF 103.  A NEW CONSTITUENT)
(*X/.1 IS PLACED BEFORE THE GIVEN NODE.  IF FAILURE, SHELVES)
(102,103,104 ARE RESTORED.)
(RESTRICTION.NO INPUT CONSTITUENT IS *X.)
(WORKSPACE.INPUT,DESTROYED. OUTPUT,SUBROUTINE RETURN.)
(SHELVES USED. 102, LEFT PART OF TREE)
(              103, MIDDLE PART OF TREE)
(              104, RIGHT PART OF TREE)
(              126, TEMPORARY STORAGE)
(              127, PUSHDOWN STORE FOR SUBROUTINES)
(EXITS. 1. SUCCESS)
(       2. FAILURE.THE NODE GIVEN IS THE INITIAL NODE.)
(          PRINT.IN SUBROUTINE DOMINATE,THE NODE TESTED,+NODE+ IS THE)
(               INITIAL NODE OF THE TREE.)
(       3. FAILURE.SHELF 103 IS EMPTY.)
(          PRINT.  IN SUBROUTINE DOMINATE, SHELF 103 IS EMPTY.)
(       4. FAILURE.THE STRING IS NOT WELL FORMED.)
(          PRINT.IN SUBROUTINE DOMINATE, THE NODE TESTED +NODE+ IS)
(               NOT DOMINATED BY ANY NODE IN SHELF 102, +LEFT PART+ .)
(THE SUBROUTINE CONTAINS 17 RULES.)
DOMINATE  $=//*A126 1,*N103 1  *
*  $1 = //*S103 1 , *A102 1  DOMINATE.1
*  $=-IN-SUBROUTINE-DOMINATE,-SHELF-*1*0*3-IS-EMPTY.*,-
     //*WAM1,*N127 1  *
*  $1=1/.3  $ (EMPTY 103)
DOMINATE.1  $1=  DOMINATE.2
*  $= //*N103 1  *
*  $=1+-IN-SUBROUTINE-DOMINATE,-THE-NODE-TESTED,-+1+-IS-THE-INITIAL-
-NODE-OF-THE-TREE.*,     //*S103 1,*WAM2,*WSM3,*WAM4,*N127 1  *
*  $1=1/.2  $ (GIVEN NODE IS ROOT)
DOMINATE.2  $1=//*S126 1  DOMINATE.2
*  $=A+*X/.0//*N126 1  *
DOMINATE.3-
   $1+*X/.L.*1 = *X/.1+1 //*S103 1,*A102 1,*S103 1 2,*X126  DOMINATE.4
*  $1+*X=1+ 2/.I1,.D.*1//*S102 1,*N126 1  DOMINATE.3
*  $ = A+B+C //*N127 1,*N103 2,*A102 3  *
* $1+$1+$ = 1/.4+2+3+-IN-SUBROUTINE-DOMINATE,-THE-NODE-TESTED,+2+-
-IS-NOT-DOMINATED-BY-ANY-NODE-IN-SHELF-*1*0*2,+3+*. //*S103 2,*Q102 3,-
*WAM4,*WSM5,*WAM6,*WSM7,*WAM8  $(NOT WELL FORMED)
DOMINATE.4  $1 = //*S102 1. DOMINATE.4
*  $ = //*N127 1  *
*  $1=1/.1  $ (SUCCESS)
(END OF SUBROUTINE DOMINATE)
```

```
(SUBROUTINE EQUAL.)
(ENTRY POINT.EQUAL)
(TEST IF THE STRINGS IN SHELVES 107 AND 108 ARE IDENTICAL.)
(RESTRICTION. NO INPUT CONSTITUENT SYMBOL IS *A.)
(WORKSPACE.INPUT,DESTROYED. OUTPUT,RETURN.)
(SHELVES USED.107 AND 108,INPUT. RESTORED IN ALL CASES.)
(              127,PUSHDOWN STORE FOR SUBROUTINES.)
(EXITS. 1. SUCCESS.EQUALITY.)
(       2. FAILURE.INEQUALITY.)
(          PRINT. IN SUBROUTINE5EQUAL,T)E STRING IN SHELF 107,+STRING+)
(            IS NOT EQUAL TO THE STRING IN SHELF 108,+ STRING.)
(SJBROUTINE EQUAL HAS 9 RULES.)
EQUAL $= *A/.1+*A// *Q107 1,*Q108 2,*N107 1,*N108 2  *
EQUAL.1  $1+1 = 2+1  EQUAL.3
*  *A+*A = A  //*N127 1  EQUAL.4
EQUAL.2  $1+$1 = 1+*A+2 //*S107 1,*S108 3,*A107 1,*A108 3 *
*  $+*A+$+*A+$+*A+$ = A+3+1+*A+7+5 //*N127 1  *
*  $1+$+*A+$ = 1/.2+2+4+-IN-SUBROUTINE-EQUAL,-THE-STRING-IN-SHELF-
-*1*0*7,*.+2+-IS-NOT-EQUAL-TO-THE-STRING-IN-SHELF-*1*0*8,*.+4+*. //-
    *Q107 2,*Q108 3,*WAM4,*WSM5,*WAM6,*WSM7,*WAM8  $ (FAILURE)
EQUAL.3  $1+1 = //*Q108 1,*Q107 2,*N107 1,*N108 2  EQUAL.1
*  $1+$1 = 2+1  EQUAL.2
EQUAL.4  $1 = 1/.1  $ (SUCCESS)
(END OF SUBROUTINE EQUAL.)
```

```
(SUBROUTINE TERMINAL.)
(ENTRY POINT.  TERMINAL)
(A TREE IS GIVEN IN THE WORKSPACE IN MODIFIED POLISH NOTATION.)
(DELETE FROM THE WORKSPACE ALL CONSTITUENTS WHICH ARE NOT TERMINAL)
(SYMBOLS. DELETE THE NUMERICAL SUBSCRIPT FROM ALL TERMINAL SYMBOLS.)
(A TERMINAL SYMBOL HAS A ZERO SUBSCRIPT OR NO NUMERICAL SUBSCRIPT.)
(A NON-TERMINAL SYMBOL HAS A NON-ZERO NUMERICAL SUBSCRIPT.)
(RESTRICTIONS.  NONE.)
(WORKSPACE.  INPUT.  ORIGINAL STRING.)
(            OUTPUT.  RETURN FOLLOWED BY TERMINAL STRING.)
(SHELVES USED.  127. PUSHDOWN STORE FOR SUBROUTINES.)
(EXIT.  SUCCESS IN ALL CASES. THE TREE NEED NOT BE WELL-FORMED.)
(THE SUBROUTINE HAS 3 RULES.)
TERMINAL  $1/.G0 = 0  TERMINAL
TERMINAL.1  $1/.0 = 1/-.  TERMINAL.1
*  $ = A+1 //*N127 1 $ (EXIT)
(END OF SUBROUTINE TERMINAL)
```